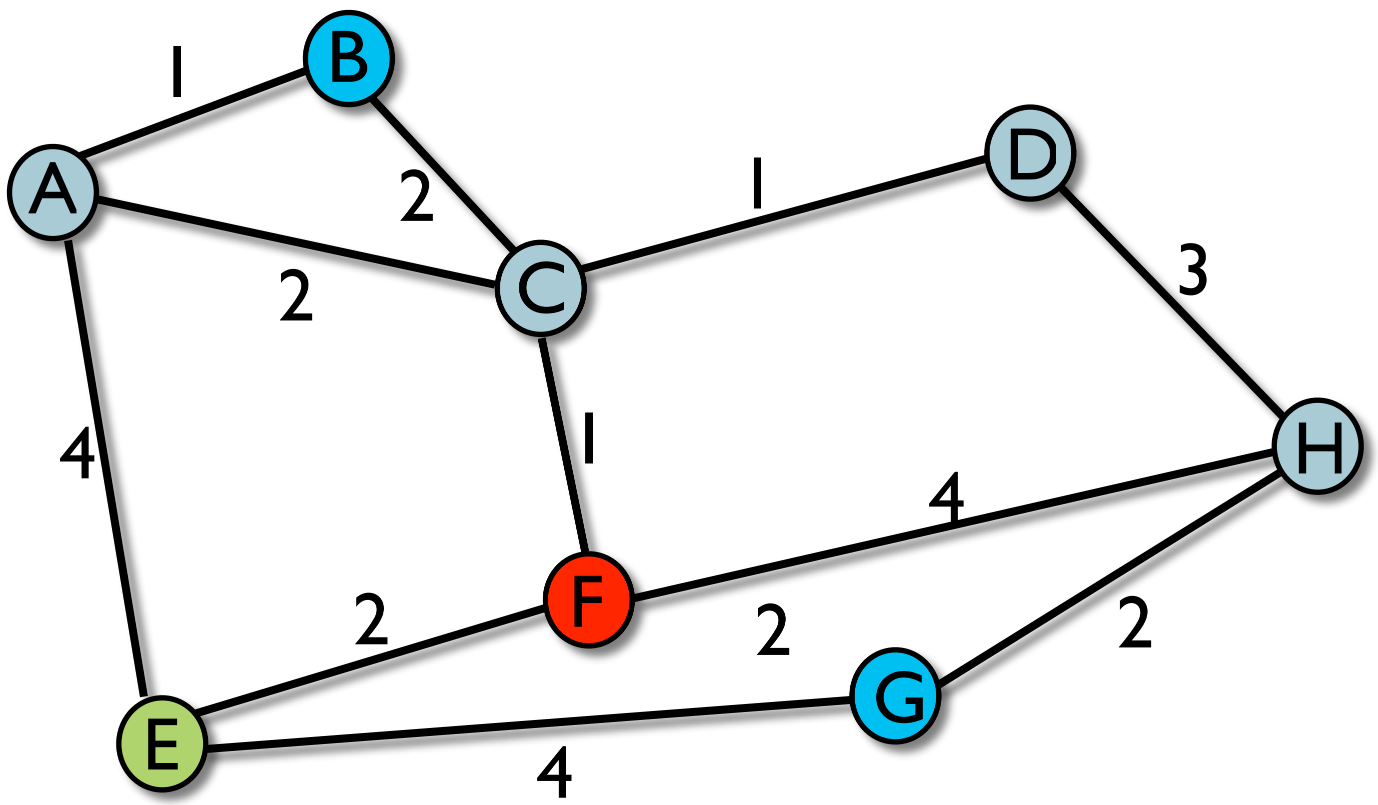
Lecture 12: Distance Oracles (ctd.)

Johannes Fischer

Reminder

- k: arbitrary parameter
- Preprocess G in $O(kn^{1/k}(n \lg n + m))$ time
  - DS of size $O(kn^{1+1/k})$ **words**
  - distance **queries** $\text{dist}_k(u,v)$ in $O(k)$ time
  - stretch $\leq 2k-1$
  - report **path** of length $\leq \text{dist}_k(u,v)$
    in $O(1)$ time per edge

Example

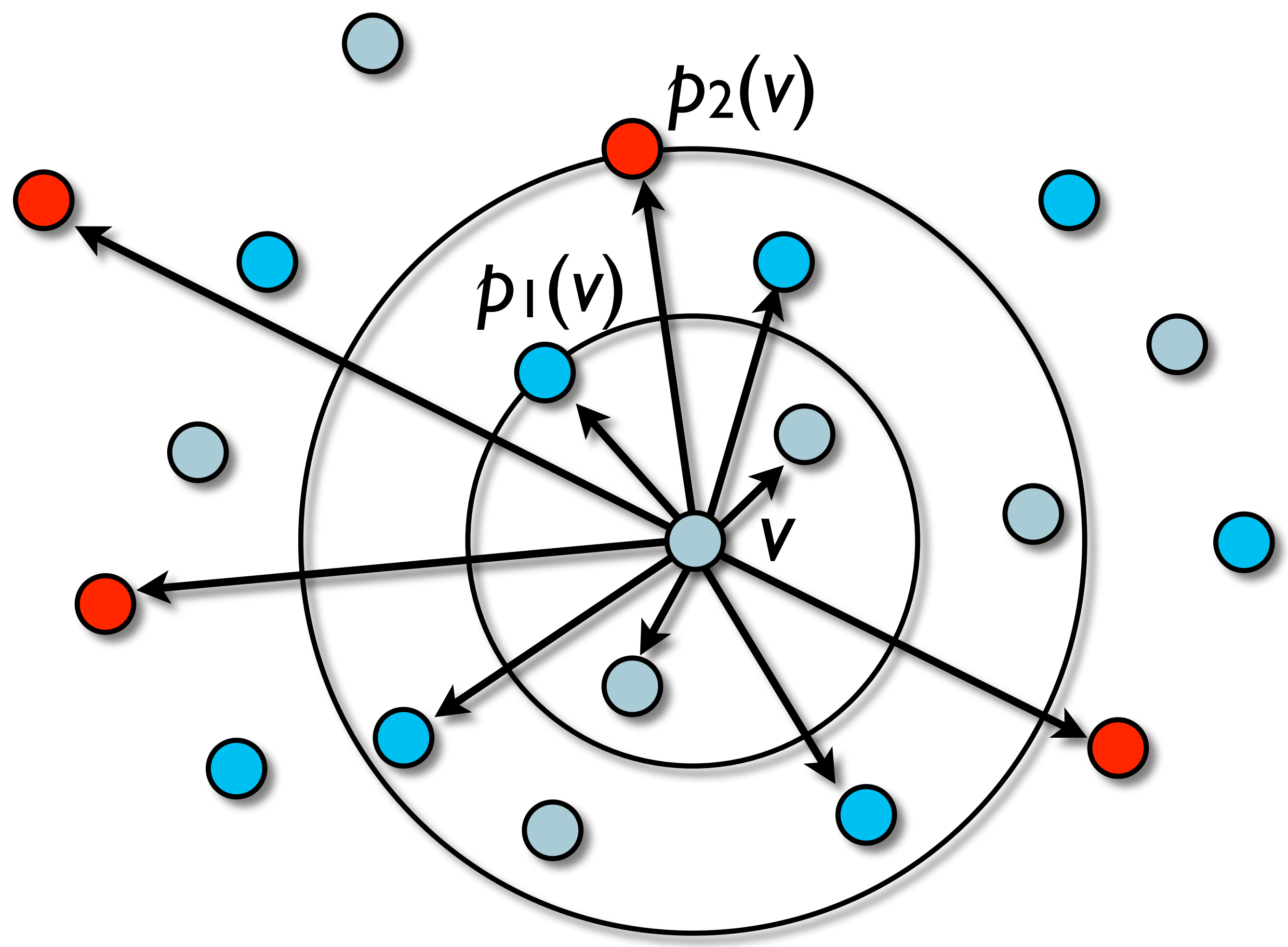


	$\delta(A_i,v)$					$p_i(v)$				
	0	1	2	3	4	0	1	2	3	4
A	0	1	3	4	∞	A	B	F	E	⊥
B	0	0	3	5	∞	B	B	F	E	⊥
C	0	1	1	3	∞	C	F	F	E	⊥
D	0	2	2	4	∞	D	F	F	E	⊥
E	0	0	0	0	∞	E	E	E	E	⊥
F	0	0	0	2	∞	F	F	F	E	⊥
G	0	0	4	4	∞	G	G	E	E	⊥
H	0	2	4	6	∞	H	G	F	E	⊥

size $O(kn)$

Bunches

- **bunch** $B(v)$ of $v \in V$:
  - $w \in B(v) \Leftrightarrow \exists i : w \in A_i \setminus A_{i+1}$ and $\delta(w,v) < \delta(A_{i+1},v)$



Query Algorithm

function dist$_k$(u,v):

$w \leftarrow u$; $i \leftarrow 0$;

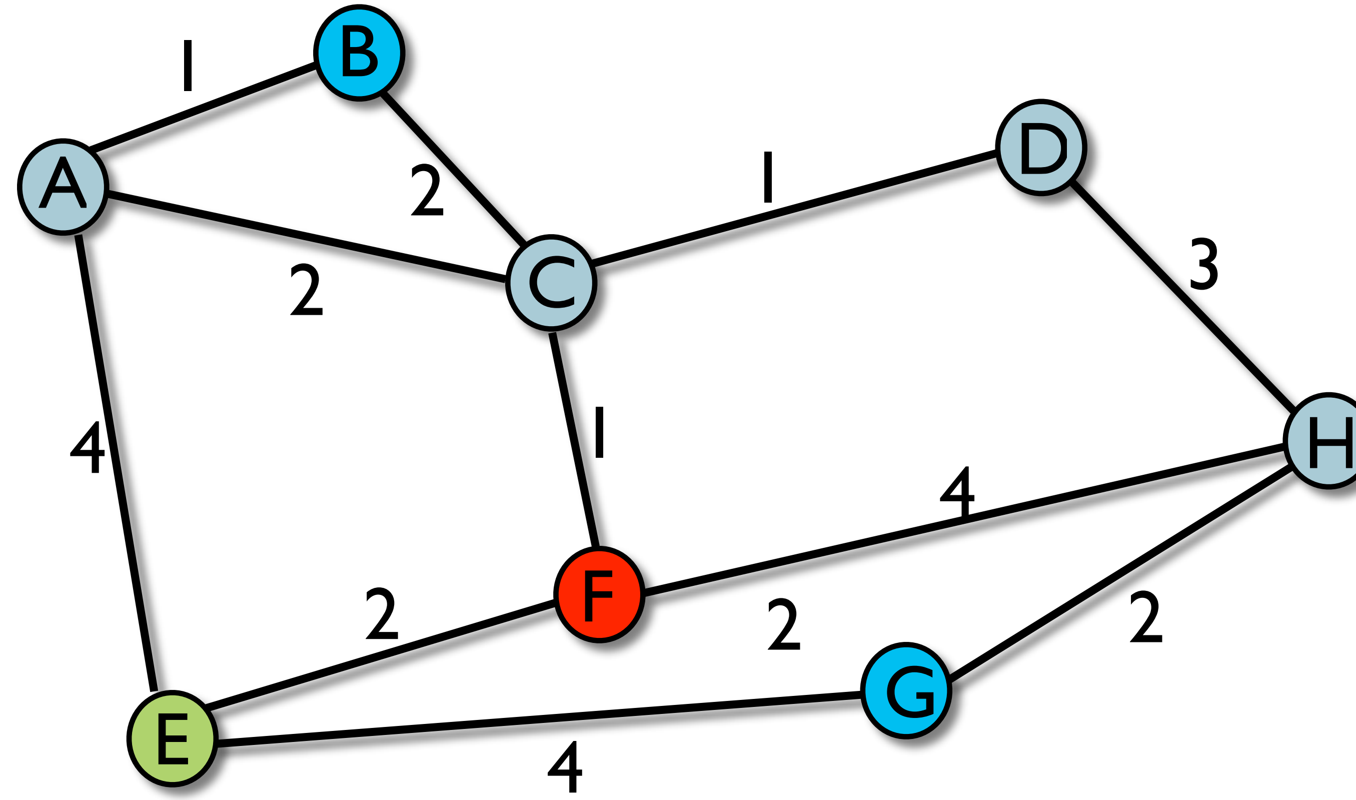
while ($w \notin B(v)$)

$i \leftarrow i+1$

$w \leftarrow p_i(v)$

$(u,v) \leftarrow (v,u)$

return $\delta(w,u) + \delta(w,v)$

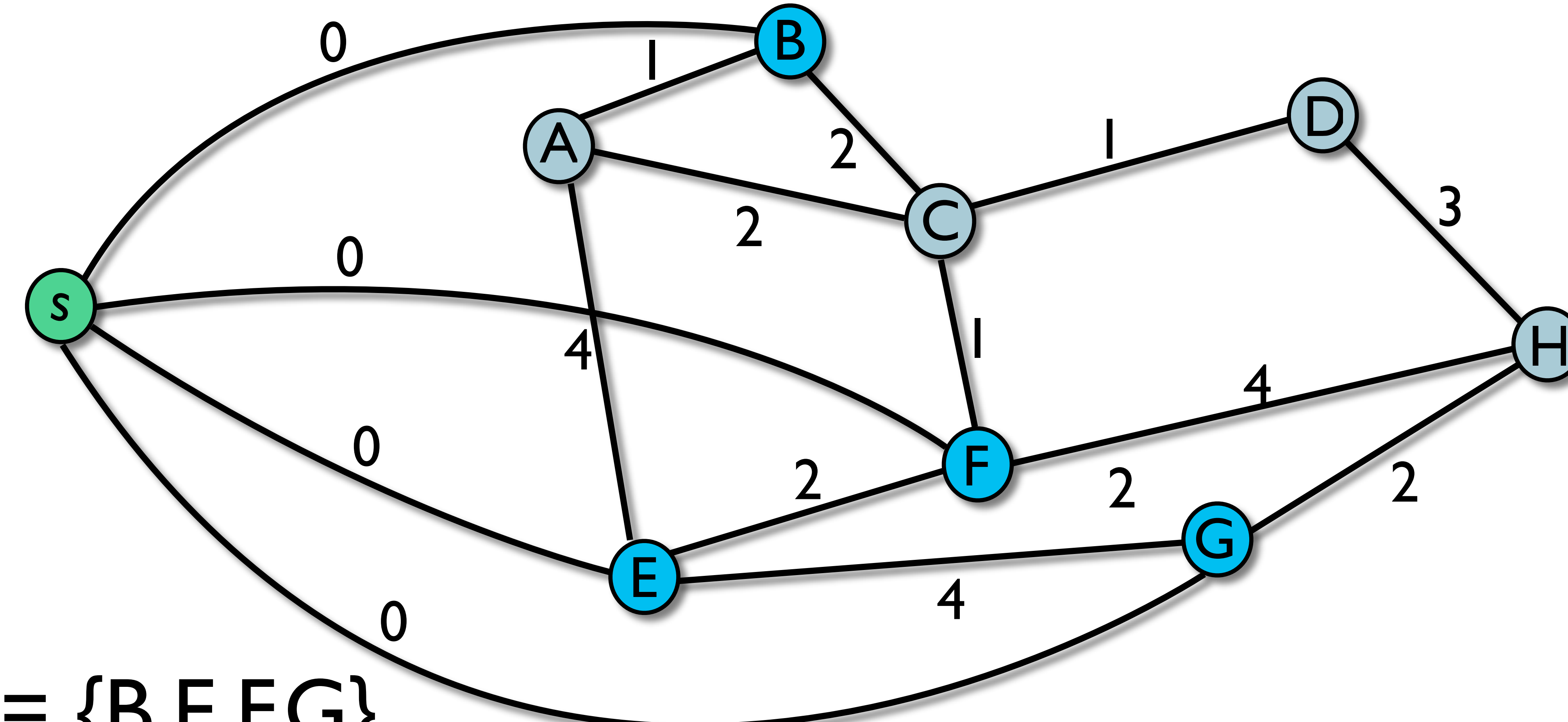


- **prove** dist$_k$(u,v) $\leq (2k-1) \cdot \delta(v,u)$
  - every iteration increases $\delta(w,u)$ by $\leq \delta(v,u) = \Delta$
  - $\Rightarrow \delta(w,u) \leq (k-1) \cdot \Delta \Rightarrow$ dist$_k$(u,v)$= \delta(w,u)+\delta(w,v) \Rightarrow$ ✓

Part II: Directly from Graphs

- last week: with distance matrix: $\mathbf{O(n^2)}$ time & (intermediate) space
- this week:
  - $\mathbf{O(kn^{1/k}(n \lg n + m))}$ time
  - $\mathbf{O(kn^{1+1/k})}$ space (space of DS)
- Main Idea:
  - compute distances by **SSSP** algorithms

Computing $\delta(A_i, v)$ & $p_i(v)$



- A_i = {B,E,F,G}
- add **new node** s to any $a \in A_i$ with $\delta(s,a)=0$
- **SSSP** from $s \Rightarrow$ distances of $v \in V$ to A_i
- time $O(n \lg n+m)$ per level $\Rightarrow O(k(n\lg n+m))$

Computing Bunches

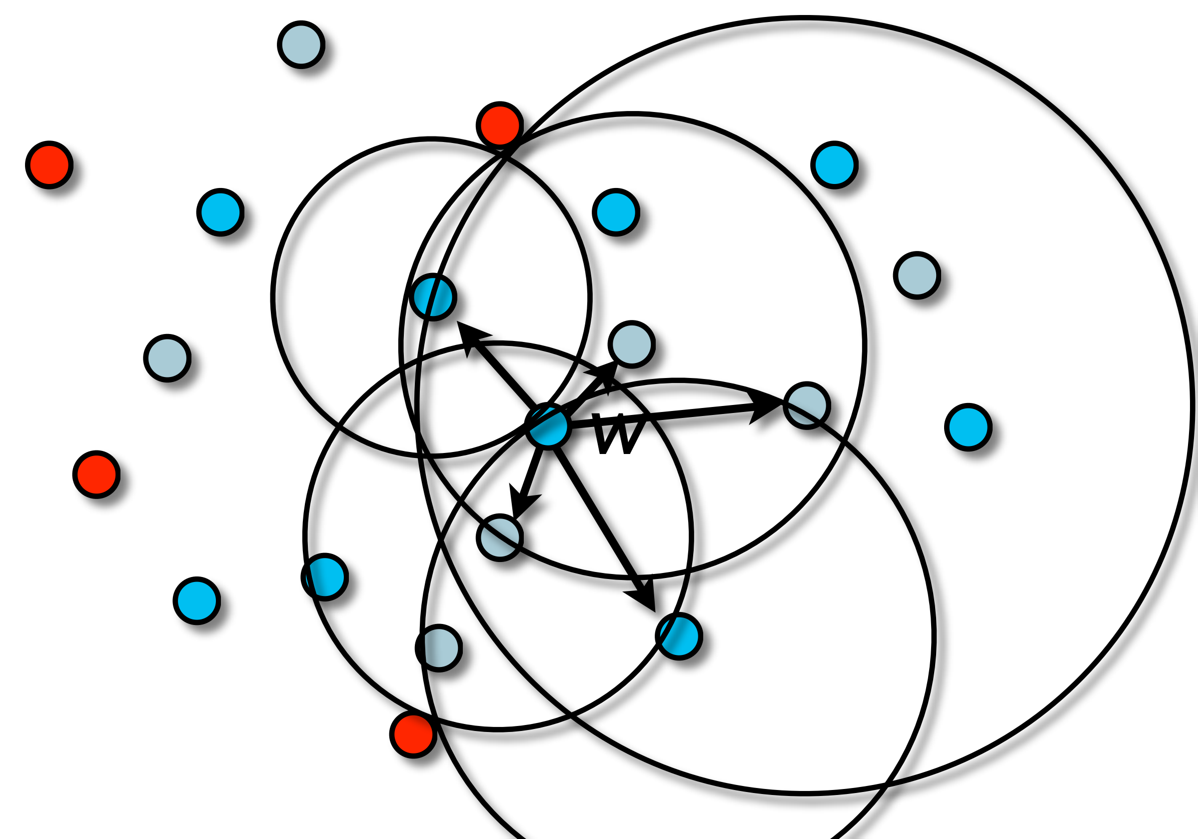
- via "inverse" of bunches called **clusters**
- $w \in A_i \setminus A_{i+1}$

  $C(w) = \{v \in V : \delta(w,v) < \delta(A_{i+1},v)\}$

$\Rightarrow v \in C(w) \Leftrightarrow w \in B(v)$

$\Rightarrow \sum|C(w)| = \sum|B(v)|$

$= kn^{1+1/k}$

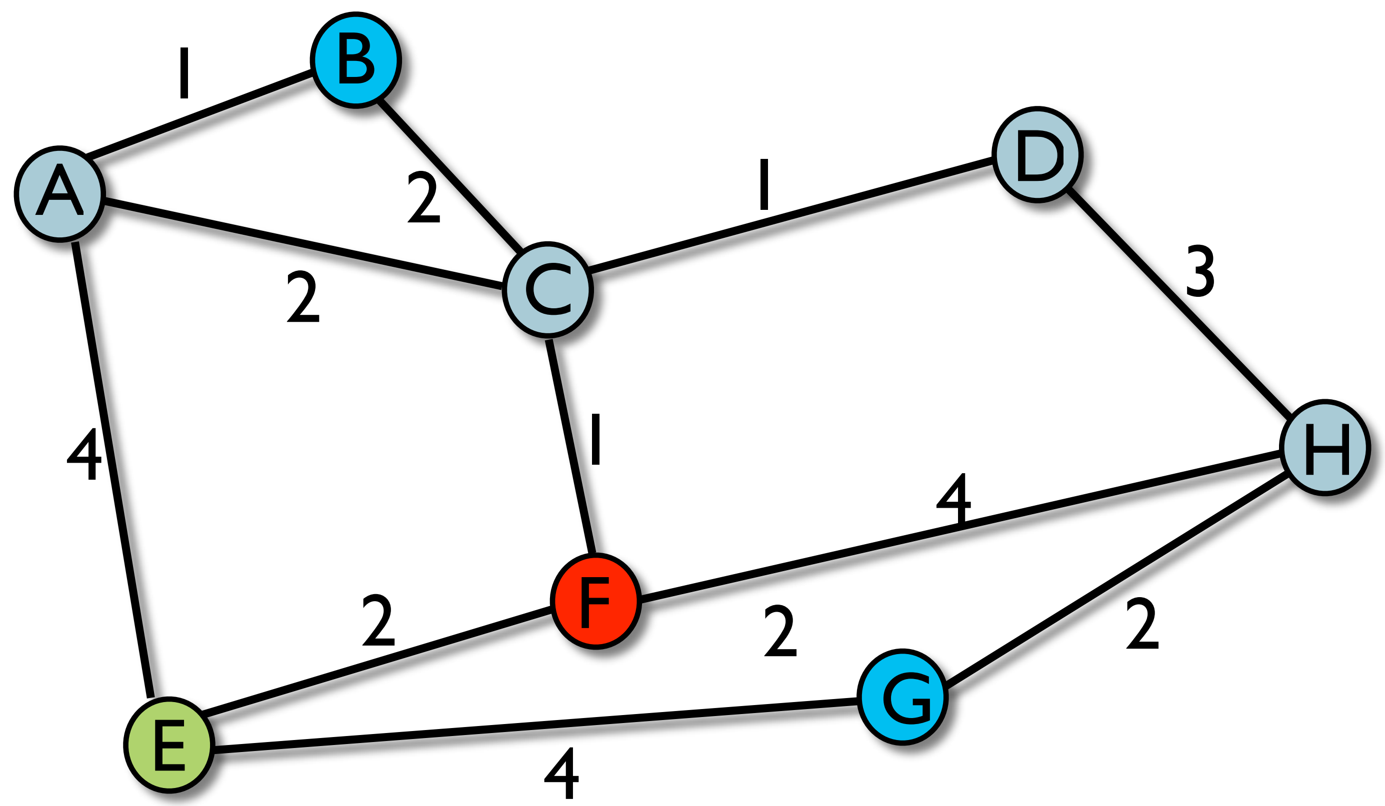


Computing Clusters

- **Dijkstra** from w:
  - maintain **estimates** $d(v) \geq \delta(w,v)$
  - **extract** $u = \text{argmin}_v\{d(v)\} \Rightarrow d(u) = \delta(w,u)$
  - **relax** (u,v) $\forall v$: $d(v) \leftarrow \min(d(v), d(u)+\omega(u,v))$
- modification for $C(w)$:
  - relax (u,v) **only if** $d(u)+\omega(u,v) < \delta(A_{i+1},v)$
- store **shortest path tree** with $C(w)$

Example

$$C(w) = \{v \in V : \delta(w,v) < \delta(A_{i+1},v)\}$$



i=2: F gets kicked out ⇒ compute C(F)

relax (u,v) **only if** $d(u)+\omega(u,v) < \delta(A_{i+1},v)$

Preprocessing Time

- $O(k(n \lg n+m))$ for the $\delta(A_i,v)$'s and $p_i(v)$'s
  - also possible in $O(knm)$ time (not covered here)
- computation of clusters:
  - $E(v)$ = edges **touching** v; $E(C(w))$ analogously
  - $\sum(|E(C(w))|+|C(w)|\lg n) = \sum|E(C(w))|+\lg n \cdot \sum|B(w)|$

$$\sum_{w \in V} |E(C(w))| \leq \sum_{w \in V, v \in C(w)} |E(v)| = \sum_{v \in V, w \in B(v)} |E(v)|$$

$$= \sum_{v \in V} |B(v)| \cdot |E(v)| = \sum_{v \in V} kn^{1/k} \cdot |E(v)| = O\left(kmn^{1/k}\right)$$

Answering Queries

- just **distance**: as before
- if also **path**:
  - finally $w = p_i(v) \in B(v)$
    $\Rightarrow v \in C(w)$
  - can also show: $w \in B(u)$
    $\Rightarrow u \in C(w)$
  - $C(w)$ stores **SP tree**
    $\Rightarrow$ return path

function $\text{dist}_k(u,v)$:
  $w \leftarrow u; i \leftarrow 0;$
  while $(w \notin B(v))$
    $i \leftarrow i+1$
    $w \leftarrow p_i(v)$
    $(u,v) \leftarrow (v,u)$
  return $\delta(w,u) + \delta(w,v)$